Combinatorial Optimization in Computer Vision
Lecture: F. R. Schmidt and C. Domokos
Exercises: T. Möllenhoff and T. Windheuser
Winter Semester 2015/2016

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 1

Room: 02.05.014
Tuesday, 27.10.2015, 14:15-15:45
Submission deadline: Tuesday, 27.10.2015, 11:15 , Room 02.09.023

Distributive Lattices (7 Points)

Exercise 1 (2 Points). Consider the set D_n of positive divisors of some natural number $n \geq 2$, partially ordered by the divisibility relation $a \preceq b \Leftrightarrow a \mid b$.

a) What are meet and join in this case? Show that D_n is a lattice.

b) Draw the Hasse diagram of D_{36}.

Exercise 2 (2 Points). Let $(\Omega, \preceq)$ be a distributive lattice, i.e.,

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z). \tag{1}$$

a) Show that if $x \preceq y$, then

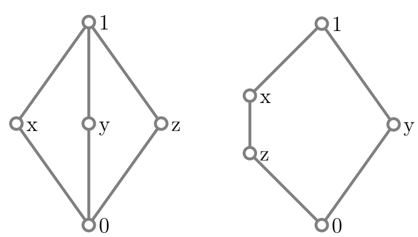
$$x \wedge y = x, \qquad x \vee y = y.$$

b) Verify that the second distributivity law,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z),$$

follows from the first one (1).

Exercise 3 (1 Point). Show that the diamond and pentagon lattices sketched below on the set $\Omega = \{0, 1, x, y, z\}$ are both non-distributive.



Exercise 4 (2 Points). Let $(\Omega, \preceq)$ be a finite totally ordered set. Give a well-defined bijective mapping $\varphi : \Omega \to \mathcal{I}_\Omega \setminus \{\varnothing\}$ such that:

$$x \preceq y \Leftrightarrow \varphi(x) \subset \varphi(y),$$

i.e., prove that there exists the following homomorphism:

$$(\mathcal{I}_\Omega \setminus \{\varnothing\}, \subset) \approx (\Omega, \preceq).$$

Submodular Functions (8 Points)

Exercise 5 (4 Points)**.** Let Ω be a finite set and let 2^Ω denote the power set of Ω. Prove the following statements:

a) The function $E : 2^\Omega \to \mathbb{R}$ defined by

$$E(A) = \sum_{i \in A} f(i),$$

is modular for any choice of $f : \Omega \to \mathbb{R}$.

b) Any modular function $E : 2^\Omega \to \mathbb{R}$ can be written as

$$E(A) = E(\varnothing) + \sum_{i \in A} \big[E(\{i\}) - E(\varnothing)\big].$$

c) Let $\Omega \subset \mathbb{Z}^2$. The function $E : 2^\Omega \to \mathbb{R}$ given as

$$E(A) = \sum_{i \in A} \sum_{\substack{j \notin A, \\ |i-j|=1}} 1,$$

is a submodular function.

d) Show that $E_h : 2^\Omega \to \mathbb{R}$ given by

$$E_h(A) := h(|A|),$$

is submodular if $h : \mathbb{R} \to \mathbb{R}$ is a concave function.
Hint: If $h : \mathbb{R} \to \mathbb{R}$ is concave, we have $h(\frac{x+y}{2}) \geq \frac{h(x)+h(y)}{2}$.

Exercise 6 (4 Points)**.** Let $\Omega = \{p, q\}$. A real-valued function on 2^Ω can be represented by $E : \{0,1\} \times \{0,1\} \to \mathbb{R}$. Let $E_1, E_2 : \{0,1\} \times \{0,1\} \to \mathbb{R}$ be two such functions, defined by the following table:

x_1	x_2	$E_1(x_1, x_2)$	$E_2(x_1, x_2)$
0	0	a	0
0	1	b	-1
1	0	c	-1
1	1	d	0

Here $a, b, c, d \in \mathbb{R}$ are constants.

a) Write down the Lovász extensions $E_1^L, E_2^L : [0,1] \times [0,1] \to \mathbb{R}$ of E_1, E_2.

b) Write down the convex closure $E_2^- : [0,1] \times [0,1] \to \mathbb{R}$ of E_2.

c) Write down the multilinear extensions $\overline{E}_1, \overline{E}_2 : [0,1] \times [0,1] \to \mathbb{R}$ of E_1, E_2.

d) Is E_2 submodular? Under which circumstances is E_1 submodular?

Hint: It is useful to a make a distinction between the cases $x_1 < x_2$ and $x_1 \geq x_2$.

Programming (4 Points)

Exercise 7 (0 Points)**.** This exercise is only necessary, if you want to program on your own computer or laptop. Download, compile and install *OpenGM* (Version ≥ 2) on your machine. The code is available from

`http://hci.iwr.uni-heidelberg.de/opengm2`

and the manual can be downloaded from

`http://hci.iwr.uni-heidelberg.de/opengm2/download/opengm-2.0.2-beta-manual.pdf`

Make sure you compile with the external libraries `MaxFlow`, `QPBO` and `TRWS`.

Make sure you have installed *Doxygen* and *cmake* on your machine. On Ubuntu just type `sudo apt-get install doxygen cmake`.

Building is a little bit tricky. You need to call `cmake` first, then `make externalLibs` and then `cmake` again. On Linux you can do this by just typing these lines on your terminal:

```
wget http://hci.iwr.uni-heidelberg.de/opengm2/download/opengm-2.3.5.
    zip
unzip opengm-2.3.5.zip
mkdir opengm-master/build
cd opengm-master/build
cmake ..
make externalLibs
cmake -DCMAKE_INSTALL_PREFIX=~/usr -DWITH_HDF5:BOOL="1" -
    DWITH_MAXFLOW:BOOL="1" -DWITH_QPBO:BOOL="1" -DWITH_TRWS:BOOL="1"
        ..
make -j 4
make doc
mkdir ~/usr
make install
```

The include files for Open GM should now be in the folder `~/usr/include/opengm`, the documentation should be located at `~/usr/doc/opengm/html` and the compiled libraries in `build/src/external`.

Exercise 8 (4 Points)**.** Install the *ImageMagick* C++ library called *Magick++*. The library is already installed on the machines in the lab. On Ubuntu you can do this by typing `sudo apt-get install libmagick++-dev`.

You can find information on Magick++ on the website

`http://www.imagemagick.org/Magick++/`

Most information you need and some easy examples are contained in the documentation of the `Magick::Image` class:

`http://www.imagemagick.org/Magick++/Image.html`

Write a program that does the following:

1. Read an image from a file.
2. Depending on the command-line arguments, perform one or more of the following operations:

   Convert the image from color to grey scale.

   Flip the x-axis of the image.

   Swap color channels.

   Display the image.
3. Save the image to a file.

The program should recognize the filenames and type of operation from the command-line arguments. For example, if the executable is called `exercise9` the call

```
./exercise9 -flipx -swaprg -display input.png output.jpg
```

would load an PNG image from the file `input.png`, flip the x-axis of the image, swap the red and green color channel, display the image and then write the result into the file `output.jpg`.